ASSIGNMENT NO.2

Title: Study of Deep learning Packages: Tensorflow, Keras, Theano and PyTorch. Document the distinct features and functionality of the packages.

Aim: Study and installation of following Deep learning Packages :

Tensor Flow

Keras

Theno iv .

PyTorch

Theory:

Deep learning is a subfield of machine learning that focuses on artificial neural networks and algorithms inspired by the structure and function of the human brain. It is a type of machine learning that uses multiple layers of interconnected nodes, often referred to as artificial neurons, to model and solve complex problems. The term "deep" in deep learning refers to the use of multiple layers in these neural networks.

Key characteristics and concepts of deep learning include:

Neural Networks: Deep learning is based on artificial neural networks, which are composed of layers of interconnected nodes. Each node, or neuron, performs a simple mathematical operation, and the network as a whole processes data through a sequence of layers.

Supervised Learning: Deep learning is often used for supervised learning tasks, where the model is trained on labeled data to make predictions or classifications. For example, it can be used for image classification, speech recognition, and language translation.

Unsupervised Learning: Deep learning can also be applied to unsupervised learning tasks, such as clustering and dimensionality reduction. In this case, the network identifies patterns and structures in the data without labeled examples.

Packages in python for supporting Machine Learning libraries:

Machine Learning Libraries:

1. Scikit-Learn: Scikit-Learn is one of the most widely used libraries for traditional machine learning tasks. It provides tools for classification, regression, clustering, dimensionality reduction, and more.
2. TensorFlow: TensorFlow is an open-source machine learning framework developed by Google. It's versatile and can be used for a wide range of machine learning tasks, including deep learning.
3. PyTorch: PyTorch is an open-source deep learning framework that is especially popular among researchers and is known for its flexibility and dynamic computation graph.

Deep Learning Libraries:

1. Keras: Keras is an open-source deep learning API that is now integrated with TensorFlow. It provides a user-friendly and high-level interface for building neural networks.
2. TensorFlow: As mentioned earlier, TensorFlow is a versatile deep learning library that can be be used for a wide range of deep learning tasks, from image and speech recognition to natural language processing.
3. PyTorch: PyTorch is also a deep learning framework that is popular for research and is known for its dynamic computation graph, making it well-suited for tasks like natural language processing and computer vision.
4. Theano: While it's less commonly used today, Theano is an older deep learning library known for its role in popularizing deep learning. It's a low-level library that served as a foundation for other libraries like Keras.

1. Developing Sequential Model:

A Sequential model in Keras is a linear stack of layers where you can add one layer at a time, starting from the input layer and progressing to the output layer. You can create a Sequential model using the `Sequential` class and then add layers using the `.add()` method.

2. Training and Validation using Inbuilt Functions:

- Keras provides high-level functions and methods to train and validate your models. You can compile a model with a loss function, an optimizer, and evaluation metrics using the `compile()` method. Then, you can train the model using the `fit()` method, specifying training data, validation data, batch size, and the number of epochs.

3. Parameter Optimization:

- Parameter optimization, also known as hyperparameter tuning, involves finding the best set of hyperparameters for your model. Keras and its ecosystem tools like KerasTuner make this

process easier. You can define a search space for hyperparameters and use KerasTuner to search for the optimal configuration.

Theano is a Python library that was designed to facilitate the development of efficient numerical computations, especially in the context of machine learning and deep learning.

While Theano is no longer actively developed as of 2017, it played a significant role in the development of deep learning frameworks and concepts. Below, I'll provide an explanation of a simple Theano program to help you understand its basic usage.

Here's a simple example of a Theano program that computes the sum of two numbers:

import theano import theano.tensor as T # Define two symbolic variables a = T.dscalar('a') b = T.dscalar('b')

Define a symbolic expression to calculate the sum c = a + b

Create a function to compile the expression add = theano.function([a, b], c) # Call the compiled function result = add(2.0, 3.0) # Print the result print("Result:", result)

The main advantage of Theano is that it can optimize and execute complex mathematical operations efficiently, taking advantage of GPU acceleration when available. While this example is quite simple, Theano's true power becomes evident when dealing with large and complex mathematical expressions, especially in the context of deep learning and neural network training.

PyTorch Tensors: PyTorch is an open-source machine learning library developed by Facebook's AI Research lab. It is widely used for various machine learning and deep learning tasks. Tensors are one of the fundamental data structures in PyTorch. Here's an explanation of PyTorch tensors:

1. Tensors as Multi-Dimensional Arrays: Tensors in PyTorch are similar to NumPy arrays but with some crucial differences. They are multi-dimensional arrays used to store and manipulate data efficiently. Tensors can have any number of dimensions, which makes them suitable for various data types, including scalars, vectors, matrices, and high-dimensional arrays.

2. Numpy-Like Interface: PyTorch tensors offer a NumPy-like interface for performing operations, but they have a significant advantage in that they can be easily moved to GPU memory, allowing for accelerated computation, especially in deep learning.

3. Creation of Tensors: You can create tensors in various ways, such as by using Python lists, NumPy arrays, or directly from data. Here's how to create a simple tensor: import torch # Create a 1D tensor a = torch.tensor([1, 2, 3]) # Create a 2D tensor b = torch.tensor([[1, 2], [3, 4]])

4. Operations on Tensors: You can perform various mathematical operations on tensors, such as addition, subtraction, multiplication, and more. These operations can be done element-wise or using standard linear algebra operations.

```
# Element-wise addition
result = a + b
# Matrix multiplication
result = torch.matmul(b, a)
```

5. Automatic Differentiation: PyTorch is known for its automatic differentiation capabilities, allowing you to compute gradients for optimization algorithms like gradient descent. This is crucial for training deep learning models.

6. GPU Acceleration: PyTorch seamlessly supports running computations on GPUs, which significantly speeds up training deep learning models. You can move tensors to the GPU using the `.to` method. # Move a tensor to the GPU a = a.to('cuda')

7. Deep Learning Integration: PyTorch is commonly used in the deep learning community and

provides high-level modules for building and training neural networks.

Uber's Pyro: Pyro is an open-source probabilistic programming library developed by Uber AI. It is built on top of PyTorch and is designed for probabilistic modeling and Bayesian inference. Here are some key features and concepts related to Pyro:

1. Probabilistic Programming: Pyro is a probabilistic programming language that allows you to express complex probabilistic models using Python code. This makes it easier to model and reason about uncertainty in data.

2. Stochastic Functions: In Pyro, you define models as stochastic functions, which include both deterministic and random variables. This allows you to build probabilistic models that capture uncertainty and dependencies in the data.
3. Inference Engine: Pyro provides an inference engine that uses various probabilistic inference algorithms, such as Variational Inference and Markov Chain Monte Carlo (MCMC), to estimate the posterior distribution over latent variables in your probabilistic model.
4. Scalable and Flexible: Pyro is designed to be scalable and flexible, making it suitable for a wide range of applications, from small-scale models to large, complex probabilistic models.
5. Applications: Pyro is used for a variety of applications, including machine learning, robotics, natural language processing, and more. It's particularly well-suited for problems where uncertainty and probabilistic modeling are essential.

Tesla Autopilot: Tesla Autopilot is an advanced driver-assistance system (ADAS) developed by Tesla, Inc. It is designed to provide semi-autonomous driving capabilities to Tesla vehicles. Here are some key points about Tesla Autopilot:

1. Features: Tesla Autopilot includes a suite of features and functionalities that enhance the driving experience. These features include adaptive cruise control, lane-keeping assistance, traffic-aware cruise control, and automated lane-keeping on highways.
2. Hardware: Tesla vehicles equipped with Autopilot come with a set of sensors and cameras, including ultrasonic sensors, radar, and a front-facing camera. These sensors are used for collecting data and enabling the vehicle to navigate and make driving decisions.
3. Self-Driving Capabilities: Tesla has introduced a feature called "Full Self-Driving" (FSD) as an extension of Autopilot. FSD aims to achieve fully autonomous driving, although it is important to note that the term "full self-driving" has been a subject of debate and regulatory scrutiny in the automotive industry. As of my last knowledge update in January 2022, FSD remains a feature that requires the driver's attention and intervention.
4. Software Updates: Tesla frequently releases over-the-air (OTA) software updates to enhance Autopilot and FSD features. These updates can add new functionalities, improve existing ones, and provide performance enhancements.
5. Driver Responsibility: It's crucial to emphasize that, as of my last update, Tesla's Autopilot and FSD systems require active supervision and intervention by the driver. Drivers are responsible for maintaining control of the vehicle and being attentive at all times.

6. Regulatory and Safety Concerns: The development and deployment of self-driving technologies, including Tesla's Autopilot and FSD, have raised regulatory and safety concerns. Different regions and countries have varying regulations and laws regarding autonomous driving, and there have been incidents and accidents associated with these technologies.

Steps/ Algorithm

Installation of Tensorflow On Ubntu:

1. Install the Python Development Environment:

You need to download Python, the PIP package, and a virtual environment. If these packages are already installed, you can skip this step.

You can download and install what is needed by visiting the following links:

https://www.python.org/

https://pip.pypa.io/en/stable/installing/ https://docs.python.org/3/library/venv.html To install these packages, run the following commands in the terminal: sudo apt update

sudo apt install python3-dev python3-pip python3-venv

Create a Virtual Environment

Navigate to the directory where you want to store your Python 3.0 virtual environment. It can be in your home directory, or any other directory where your user can read and write permissions.

mkdir tensorflow_files cd tensorflow_files

Now, you are inside the directory. Run the following command to create a virtual environment:

python3 -m venv virtualenv

The command above creates a directory named virtualenv. It contains a copy of the Python binary, the PIP package manager, the standard Python library, and other supporting files. Activate the Virtual Environment source virtualenv/bin/activate

Once the environment is activated, the virtual environment's bin directory will be added to the beginning of the $PATH variable. Your shell's prompt will alter, and it will show the name of the virtual environment you are currently using, i.e. virtualenv. Update PIP pip install --upgrade pip

5. Install TensorFlow

The virtual environment is activated, and it's up and running. Now, it's time to install the TensorFlow package.

pip install -- upgrade TensorFlow

Installation of Keras on Ubntu :

Prerequisite : Python version 3.5 or above.

STEP 1: Install and Update Python3 and Pip

Skip this step if you already have Python3 and Pip on your machine**.** sudo apt install python3 python3.pip sudo pip3 install —upgrade pip STEP 2: Upgrade Setuptools pip3 install —upgrade setuptools

STEP 3: Install TensorFlow pip3 install tensorflow

Verify the installation was successful by checking the software package information: pip3 show tensorflow

STEP 4: Install Keras pip3 install keras

Verify the installation by displaying the package information:

pip3 show keras

[https://phoenixnap.com/kb/how-to-install-keras-on-linux]

Installation of Theano on Ubuntu:

Step 1: First of all, we will install Python3 on our Linux Machine. Use the following command in the terminal to install Python3.

sudo apt-get install python3 **Step 2**:

Now, install the pip module *s*udo

apt install python3-pip **Step 3**:

Now, install the Theano

Verifying Theano package Installation on Linux using PIP python3 -m pip show theano

Installation of PyTorch

First, check if you are using python's latest version or not.Because PyGame requires python 3.7 or a higher version python3 –version

pip3 –version pip3 install torch==1.8.1+cpu torchvision==0.9.1+cpu torchaudio==0.8.1 -f https://download.pytorch.org/whl/torch_stable.html

[Ref : https://www.geeksforgeeks.org/install-pytorch-on-linux/]

Python Libraries and functions required

Tensorflow,keras numpy : NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy stands for Numerical Python. To import numpy use import numpy as np

pandas: pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. To import pandas use import pandas as pd

sklearn : Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. For importing train_test_ split use

from sklearn.model_selection import train_test_split

For TheaonRequirements:
Python3
Python3-pip
NumPy
SciPy
BLAS

Code:

Tensorflow Test program:

```
import tensorflow as tf

print(tf.__version__)

2.1.0

print(tf.reduce_sum(tf.random.normal([1000, 1000])))

tf.Tensor(-505.04108, shape=(), dtype=float32)
```

Keras Test Program:

```
from tensorflow import keras from
keras import datasets

# Load MNIST data #
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data() #
# Check the dataset loaded # train_images.shape,
test_images.shape
Theano test program
# Python program showing # addition of two scalars

# Addition of two scalars import numpy import
theano.tensor as T from theano import function

# Declaring two variables x = T.dscalar('x') y
= T.dscalar('y')

# Summing up the two numbers z = x + y

# Converting it to a callable object
# so that it takes matrix as parameters
```

f = function([x, y], z) f(5, 7)
Test program for PyTorch

The usual imports import

torch import torch.nn as nn

print out the pytorch version used print(torch. version)

Output of Code:

```
In [1]:
1 import tensorflow as tf

C:\Users\DELL\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning:
ed for this version of SciPy (detected version 1.26.0
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"

In [2]:
1 print(tf.__version__)

2.14.0

In [3]:
1 print(tf.reduce_sum(tf.random.normal([1000,1000])))

tf.Tensor(-1920.7336, shape=(), dtype=float32)

In [5]:
1 from tensorflow import keras
2 from keras import datasets
3
```

Figure 1: Output of tensoflow

The above figure shows the importing of tensorflow.

```
5]:
1 (train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data() #
2 # Check the dataset loaded #
3 train_images.shape, test_images.shape
4

)ut[6]: ((60000, 28, 28), (10000, 28, 28))
```

Figure 2: Output of tensoflow

The above figure shows the checking of dataset loaded.

Conclusion :

Tensorflow , PyTorch,Keras and Theano all these packages are installed and ready for Deep learning applications . As per application domain and dataset we can choose the appropriate package and build required type of Neural Network.